In the Claims

This listing of claims will replace all prior versions and listings of claims in the application:

1. (Original) A method for maintaining coherency of software breakpoints in shared memory when debugging a multiple processor system, the method comprising the steps of:

activating a first debug session associated with a first processor of a plurality of processors and at least a second debug session associated with a second processor of the plurality of processors;

setting a first software breakpoint in a shared memory location in the first debug session such that all debug sessions are notified of the setting of the breakpoint; and

clearing the first software breakpoint in the shared memory location in the second debug session such that all debug sessions are notified of the clearing of the breakpoint.

2. (Original) The method of Claim 1 wherein the method comprises the step of creating a software memory map of the memory usage of a plurality of processors in the system to be debugged.

3. (Original) The method of Claim 2 wherein the step of setting comprises:

searching the software memory map to find a first plurality of processors having read access to the shared memory location;

updating a software representation maintained for software breakpoints for each of the first plurality of processors; and

writing the software breakpoint instruction in the shared memory location.

4. (Original) The method of Claim 3 wherein the step of writing comprises a method for selecting a processor to execute the write, the method comprising the steps of:

if the first processor associated with the first debug session requesting the setting of the software breakpoint has write access to the shared memory location

then

selecting the first processor to perform the write request;

else performing the following steps a-b:

a. searching the software memory map for a second processor with write access to the shared memory location;

b. selecting the second processor to perform the write request; and

passing the software breakpoint instruction to the selected processor to be written into the shared memory location.

5. (Original) The method of Claim 4 wherein the step of passing the software breakpoint instruction comprises the steps of:

searching the software memory map for a second plurality of processors that have read access to the shared memory location;

broadcasting the write request to the second plurality of processors; and

performing cache coherency updates in response to the write request in each of the second plurality of processors as required.

6. (Original) The method of Claim 5 wherein the step of broadcasting the write request comprises indicating that the write request is intended for maintaining cache coherency as opposed to a normal write request.

7. (Original) The method of Claim 6 wherein the step of performing comprises using cache coherency capabilities, if any, of a processor in response to the write request intended for maintaining cache coherency.

8. (Original) The method of Claim 2 wherein the step of clearing comprises:

writing the original instruction stored in a software representation maintained for software breakpoints into the shared memory location;

searching the software memory map to find a fourth plurality of processors having read access to the shared memory location; and

updating a software representation maintained for software breakpoints for each of the fourth plurality of processors to remove the software breakpoint for the shared memory location.

9. (Original) The method of Claim 8 wherein the step of writing comprises a method for selecting a processor to execute the write, the method comprising the steps of:

if the second processor associated with the second debug session requesting the clearing of the software breakpoint has write access to the shared memory location

then

selecting the second processor to perform the write request;

else performing the following steps a-b:

a. searching the software representation of the memory map for a third processor with write access to the shared memory location;

b. selecting the third processor to perform the write request; and

passing the original instruction to the selected processor to be written into the shared memory location.

10. (Original) The method of Claim 9 wherein the step of passing the original instruction comprises the steps of:

searching the software memory map for a fifth plurality of processors that have read access to the shared memory location;

broadcasting the write request to the fifth plurality of processors; and

performing cache coherency updates in response to the write request in each of the fifth plurality of processors.

11. (Original) The method of Claim 10 wherein the step of broadcasting the write request comprises indicating that the write request is intended for maintaining cache coherency as opposed to a normal write request.

12. (Original) The method of Claim 11 wherein the step of performing comprises using cache coherency capabilities, if any, of a processor in response to the write request intended for maintaining cache coherency.

13. (Original) The method of Claim 1 comprising the step of stepping over a software breakpoint or running after hitting a breakpoint in a shared memory location wherein the method for stepping over the software breakpoint comprises the steps of:

requesting that a software breakpoint in a shared memory location be stepped over or program execution resumed after hitting the breakpoint in a third debug session;

clearing the software breakpoint in the shared memory location in the third debug session such that all debug sessions are notified of the clearing of the breakpoint;

stepping a processor associated with the third debug session to the instruction after the shared memory location from which the software breakpoint was cleared; and

setting the first software breakpoint in the shared memory location in the third debug session such that all debug sessions are notified of the setting of the breakpoint.

14. (Currently Amended) The method of Claim 13 additionally comprising the steps of:

searching ~~the~~ a software memory map for a sixth plurality of processors having read access to the shared memory location; and

halting all processors in the sixth plurality of processors after the step of requesting and prior to the step of clearing.

15. (Currently Amended) The method of Claim 13 wherein the step of setting comprises:

searching ~~the~~ a software memory map to find a seventh plurality of processors having read access to the shared memory location;

updating a software representation maintained for software breakpoints for each of the seventh plurality of processors; and

writing the software breakpoint instruction in the shared memory location.

16. (Currently Amended) The method of Claim 13 wherein the step of clearing comprises:

writing the original instruction stored in a software representation maintained for software breakpoints in the shared memory location;

searching ~~the~~ a software memory map to find an eighth plurality of processors having read access to the shared memory location; and

updating a software representation maintained for software breakpoints for each of the eighth plurality of processors to remove the software breakpoint for the shared memory location.

17. (Original) A software development system, comprising:

a memory storage system holding a software development tool program;

a host computer connected to the memory storage system, the host computer operable to execute the software development tool program;

a test port for connecting to a target hardware system, the hardware system being comprised of multiple processors with common shared memory and operable to execute an application program; and

wherein the software development tool is operable to support debugging of the application program executing on the target hardware system using a method for maintaining coherency of software breakpoints in shared memory when debugging a multiple processor system, the method comprising the steps of:

activating a first debug session associated with a first processor of a plurality of processors and at least a second debug session associated with a second processor of the plurality of processors;

setting a first software breakpoint in a shared memory location in the first debug session such that all debug sessions are notified of the setting of the breakpoint; and

clearing the first software breakpoint in the shared memory location in the second debug session such that all debug sessions are notified of the clearing of the breakpoint.

18. (Original) The software development system of Claim 17 wherein the method comprises the step of creating a software memory

map of the memory usage of a plurality of processors in the system to be debugged.

19. (Original) The software development system of Claim 18 wherein the step of setting comprises:

searching the software memory map to find a first plurality of processors having read access to the shared memory location;

updating a software representation maintained for software breakpoints for each of the first plurality of processors; and

writing the software breakpoint instruction in the shared memory location.

20. (Original) The software development system of Claim 19 wherein the step of writing comprises a method for selecting a processor to execute the write, the method comprising the steps of:

if the first processor associated with the first debug session requesting the setting of the software breakpoint has write access to the shared memory location

then

selecting the first processor to perform the write request;

else performing the following steps a-b:

a. searching the software memory map for a second processor with write access to the shared memory location;

b. selecting the second processor to perform the write request; and

passing the software breakpoint instruction to the selected processor to be written into the shared memory location.

21. (Original) A digital system, comprising:

multiple processors with common shared memory for executing an application program; and

wherein the application program was developed with a software development system using a method for maintaining coherency of software breakpoints in shared memory when debugging a multiple processor system, the method comprising the steps of:

activating a first debug session associated with a first processor of a plurality of processors and at least a second debug session associated with a second processor of the plurality of processors;

setting a first software breakpoint in a shared memory location in the first debug session such that all debug sessions are notified of the setting of the breakpoint; and

clearing the first software breakpoint in the shared memory location in the second debug session such that all debug sessions are notified of the clearing of the breakpoint.

22. (Original) The digital system of Claim 21 wherein the method comprises the step of creating a software memory map of the memory usage of a plurality of processors in the system to be debugged.

23. (Original) The digital system of Claim 22 wherein the step of setting comprises:

searching the software memory map to find a first plurality of processors having read access to the shared memory location;

updating a software representation maintained for software breakpoints for each of the first plurality of processors; and

writing the software breakpoint instruction in the shared memory location.

24. (Original) The digital system of Claim 23 wherein the step of writing comprises a method for selecting a processor to execute the write, the method comprising the steps of:

if the first processor associated with the first debug session requesting the setting of the software breakpoint has write access to the shared memory location

then

selecting the first processor to perform the write request;

else performing the following steps a-b:

a. searching the software representation of the memory map for a second processor with write access to the shared memory location;

b. selecting the second processor to perform the write request; and

passing the software breakpoint instruction to the selected processor to be written into the shared memory location.